Application of Machine Learning for Test Case Optimization in Functional Regression Testing of GUIs: Exploring the Current State

Sara Khan
Dept. of Computer Application
Veer Bahadur Singh PurvanchalUniversity
Jaunpur, Uttar Pradesh, 222003, India

Saurabh Pal
Dept. of Computer Application
Veer Bahadur Singh Purvanchal University
Jaunpur, Uttar Pradesh, 222003, India

ABSTRACT
This paper tries to explore recent research developments in the application of Machine Learning in functional regression testing of GUIs, mainly focusing towards test case optimization scenarios. A brief literature study was conducted by exploring the available literature from top digital repositories mainly from years 2017-2022 and identifying the research gaps and challenges. Analysis reported certain important research gaps in the available literature and also challenges faced by researchers. This paper provides a quick overview for those who are interested in this area of research. Simplified description and presentation of the research literature provides clear mapping for further research scope.

General Terms
Software Testing, Machine Learning, Automation

Keywords
GUI, Test Case Optimization, Functional Regression Testing

1. INTRODUCTION
Testing of Graphical User Interface (GUI) has become a very complex task, since interface of today's application is not simple. As the interdependence of different components of the GUI with one another and with the environment has increased, it has become a challenge for testers to perform timely and accurate testing of these GUIs. Increasing adoption of Agile and CI/CD environment in the organization have made it more intense process to be performed. As the software evolves, changes are integrated in the software, which directly imposes certain level of threat to the existing software test suites. To ensure that no threat is introduced to the available software component functionality, Regression Testing is performed. If simply defined, Regression Testing is a form of testing that ensures all the functionalities of software work correctly after required changes are incorporated. This testing can be performed in three ways i.e. partial, selective and all [1]. Selection entirely depends upon the situation and impact of the changes. Since, Machine Learning (ML) is a recent area of research in almost every dimension. Its application in handling GUI regression testing is an important prospect in terms of attaining quality and saving human effort in an organization. GUI testing revolves around mainly two types of testing i.e. Behavioral or Functional Testing and Structural or Non-Functional testing. Functional testing is a black box testing technique that is use to validate the behavior of the GUI in response to interaction with the users. It is use to inspect entire functionality of the UIX components like validating the functioning of login page, hyperlinks, and submit buttons etc. Whereas, Non-Functional testing deals with internal implementation like layout, navigation, appearance etc. Both types of testing brings huge challenge for the testers since, GUI's are susceptible to changes in accordance to the browsers they run on, operating systems, screen size, resolution etc. Creating and maintaining the test scripts require huge efforts and cost. Even a slight change may require complete updating in the test cases. Hence, ML and Deep Learning (DL) has started playing a huge role in providing a more feasible solution to perform test case creation and maintenance [2]. This paper dives to study the same through ML only .It is arranged as follows. Sec. 2 describes research objectives, Sec. 3 describes challenges in GUI regression testing, Sec. 4 has related literature study, Sec. 5 enlists the identified research gaps and Sec. 6 provides conclusion to the paper

2. RESEARCH OBJECTIVES
Main objective of this research is to study the research development related to GUI functional regression testing. It can be further stated as below.

RO1: *To study the challenges of GUI functional regression testing in organizations.*
RO2: *To study the latest available literature for ML related methodologies in test case optimization w.r.t regression testing.*
RO3: *To document the research gaps.*
RO4: *To provide the future prospects of research.*

3. STUDY ON CHALLENGES OF GUI FUNCTIONAL REGRESSION TESTING
As companies are struggling to meet user expectations on the available resources, automation of testing is an integral process in an organization to speed up the release cycle. Researchers has studied the challenges of GUI test automation through important 49 publications [3]. It reported 24 core challenges which was categorized into 3 categories namely challenge in Test Execution Fragility, Appropriate Tools and Automation Skills and Model-Based Testing (MBT).Though, GUI automation is considered to be the toughest, with the application of Agile software development model, testing process has started switching towards AI- Enabled tools to support the process with much accuracy and least cost, like Static code analyser 'Facebook Infer' [4], Unit test generation 'Diffblue' [5] and entire testing platform 'SmartBear' [6]. A study in [7] stated that 80% of maintenance cost comes from regression testing and that too maximum effort is due to testing of GUIs. Goal is to reduce regression cycle and also technical and financial cost. The challenge is to attain robust automation and capture the bugs before production. Issues related with GUI regression testing can be categorized into

three core categories that can be understood further through Table 1.

Table 1.Key challenges in GUI regression testing

Automation Process	Issues/Challenges
GUI Test Case Designing	Test Case Generation (TCG), Test Framework Designing, Risk Based Profiling
GUI Test Case Maintenance	Test Case Repair (TCR)
GUI Test Execution	Decision on type of regression testing to execute (all, partial, hybrid or prioritized), Test Case Optimization (selection and minimization)

3.1 GUI Test Case Designing

It is the process of deciding testing scenarios, framework and generation of test cases for testing the GUIs. Goal of Test Case Designing is to maximize the test coverage. Core techniques used for designing is generally Boundary Value Analysis, Equivalence Class partitioning, Decision Table, State Transition and Error Guessing. Details of these techniques can be read at [8]. Some of the important literature who contributed to address this issue recently has been provided in Table 2.

Table 2. Important literature who contributed towards GUI TCG for regression testing in the past 5 years.

Ref.	Contribution	Methodology
Pradeep Kumar, Rajesh Bhatia [9]	Agent Based approach to generate regression test cases.	Combined UML case diagram, use cases and activity diagram to identify changes at both syntax and semantic levels.
Thomas Wetzlmaier and Rudolf Ramler [10]	Hybrid approach combining available regression tests and random test generation technique.	Based on "monkey testing framework". Random interaction with the GUI is done and SUT behaviour is recorded to detect failures.
M. Medhat Kamal, SaadM.Darwish, Ahmed Elfatatry [11]	TCG of GUI for web application using HTML file.	Test cases are generated individually for the elements of the web page, and also between different elements.
Granda, M., Parra, O. and Alba-Sarango, B [12].	TCG using user requirement specification.	Based on "Agile-Model driven development". Test scenarios are generated by parsing user stories.

3.2 GUI Test Case Maintenance

As the software grow from time to time, its complexity increases. Therefore, it becomes a necessity to efficiently capture, retrieve, execute and store the test cases, so to avoid any unnecessary human efforts that can delay scheduled delivery. Starting from test case breakage to test case management, GUI testing possess certain challenges at software evolution. Latest research related to test case breakage and repair in case of software evolution is mentioned in Table 3.

Table 3. Research w.r.t. GUI TCR for regression testing

Ref.	Objective	Contribution
AtifMemon, Mary Lou Soffa [13]	Test Case Generation and Repair w.r.t regression testing of GUIs.	Divides the available regression test cases into usable and unusable according to changes. Unusable ones are further repaired according to new changes using event sequence.
Minxue Pan, Tongtong Xu, Yu Pei , Zhong Li, Tian Zhang andXuandong Li[14]	GUI test scripts repair for regression testing.	Proposed an approach 'METER' that uses computer vision for GUI changes. It works on screenshots to infer GUI changes and repairmen.
Zebao Gao, ZhenyuChen,YunxiaoZou and Atif M. Memon [15]	GUI test scripts repair for regression testing.	Proposes novel approach 'SITAR' for repairing of low level test script using three main steps i.e. Ripping, Mapping and Repairing.

3.3 GUI Test Case Execution

Major challenge lies in deciding which all test cases need to be executed for regression testing. Since, executing all the test cases is infeasible w.r.t time and cost, therefore certain methods like Test Case Prioritization (TCP), Test Case Selection (TCS) and Test Case Minimization (TCM) of the

test cases can be applied on the basis of relevancy of test cases for the applied changes. Importance of TCS has been seen effective in real scenarios. Detailed analysis of the available literature on TCS has been done by [16], covering from year 1997 to 2006.Parameters forvaluation was cost reduction and fault detection effectiveness. Another research review has been done by [17], from 2007 to 2015. Result observed that maximum of the studies being analyzed, used cost as the effectiveness measure compared to fault-detection capability and coverage, with coverage being minimum. TCP is considered as the most efficient method for executing test cases, as it does not eliminate any test cases, instead, it prioritizes the test cases based on certain weights. TCM method eliminates unnecessary test cases which no longer is related to the applied changes.It is one of the process for Test Case Reduction. Figure 1 demonstrates current scenario of research done for Test Case Optimization (TCO) based on TCG, TCS, TCM, TCR and TCP.

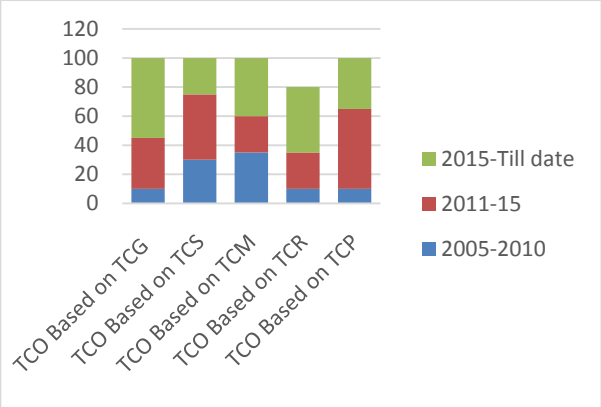

Figure 1. Literature Count for Test Case Optimization

4. LITERATURE STUDY ON ML BASED TEST CASE OPTIMIZATION

Question arises *what can be different possibilities in which ML can be applied to reduce the cost of regression testing*. By studying various literature, it was found that academic researchers are generally inclined towards optimizing the test cases through techniques of prioritization, selection and reduction/pruning/minimization. Whereas, researchers from industries are more tending their focus on doing optimization by determining obsolete test cases, unreliable test cases and flaky test cases. But the goal remains same i.e. better code coverage, improved traceability and better fault prediction. Authors in [18] proposed an optimization model based on historical data of test cases. They tried to provide the solution to existing optimization techniques which mainly rely on source code dependency. In this study, aim is to explore the application of ML in this context. We gathered the literature from different reputed repositories like IEEE Explore, ACM Digital library, Scopus, Web of Science, Google Scholar, Wiley online libraries and others. Greatest challenge was to retrieve exact literature concerning to our problem statement. Different queries were framed like<ML, GUI Testing, and Regression Testing>, <Automation, UI Test Cases, and Optimization>, < GUI Test Cases, AI, Regression Testing> and many more, but it was found that such a literature addressing all the keywords in our problem statement were very rare. Though, research papers using traditional methods of natural optimization methods like ABCO (Ant Bee Colony Optimization), PSO (Particle Swarm Optimization), Bat Optimization algorithm, etc. were huge, but papers regarding ML in this context were very limited. Secondly, it was found that objective for optimization of the GUI test cases were generally limited for improving the test coverage, reducing the feedback time and improving the fault detection ability.

4.1 Optimizing Regression TCS and TCP Using ML

Selecting the desired test case based on the changes is desirable to reduce the cost of regression testing. Engineers of Meta team has proposed a method called 'Predictive Test Selection' [19]. Their objective is to determine the likelihood of a given test to be able to find the regression. Method allows training the ML model using historical code changes. Gradient-boosted machine learning model has been used for training. Another methodology based on 'Test Impact Analyses' creates an adaptive subset [20]. Its objective lies in selecting the subsets of test cases by applying ML model on the test execution data. Data being information related to Git commits and test execution results. Retrieving the literature related to our topic, it was found that very less literature in the past 5-6 years is available i.e. less than 20 papers can be found that are directly or indirectly related to optimization of GUI test cases in regression testing, though huge amount of TCP and TCS literature can be found using traditional optimization techniques in general, like prioritization through social network analysis, code coverage, multi-objective prioritization etc. [21]-[23]. Table 4 provides overview of some of the important and recent research work carried out in this area.

Table 4. TCP of GUI/UI test cases

Ref.	Latest Research Work
[24]	***Name of Technique/Methodology:*** TERMINATOR ***Research Objective:*** Better Automated UI Test Case Prioritization technique ***Research Gap addressed:*** Fewer 'Black Box' based TCP approach ***ML Model:*** Support Vector Machine(SVM) ***Feature Set:*** Execution history, Test Case Description, Feedback ***Metric:*** APFDc (Average percentage of faults detected with cost) ***Study:*** This approach predicts which tests might fail sooner for a fault. Since, it has been studied that UI test cases failure do not follow a particular pattern. This classification cannot be always accurate in real situations.

[25]	***Name of Technique/Methodology:*** Scripted GUI Testing of Android Apps ***Research Objective:*** Analyse GUI Test Cases Fragility in case of software evolution. ***Research Gap addressed:*** Provided a practical analyses of Android Testing Framework and some important GUI tools on Test Suite Evolution, Fragility, Diffusion. ***Study:*** Results concluded that on average, 7.5% of the changed lines are in the GUI test code and 3% of the test code is modified. And in terms of fragility, on average 1 for 5 classes in each test suite needs modifications.
[26]	***Name of Technique/Methodology:*** RLTCP: Reinforcement Learning approach to prioritizing automated user interface tests. ***Research Objective:*** To prioritize the test cases. ***Research Gap addressed:*** To reduce the need for full execution. ***ML Model:*** Reinforcement Learning. ***Feature Set:*** Execution History, Coverage Weights. ***Metric:*** APFD (Average percentage of faults detected) ***Study:*** Results concluded that RLTCP outperformed other methods of TCP i.e. Original order, Random order, History based randomised (HBRF), RETECS. Result showed significant differences in APFD values. RLTCP and HBRL (History Based Randomised new tests Last) performed same with no significant difference in APFD values.

4.2 Test Suite Reduction using ML

Test Suite Reduction or Test Case Minimization is the process of reducing the test cases by retaining the coverage and fault detection ability. Main objective is to minimize the test suite so that test case maintenance cost can be reduced. Many researchers have tried to reduce the test suite size by identifying redundant, flaky and obsolete test cases. Different other methods have been identified that helps in reduction keeping in view the coverage and fault detection capability. Authors of [27] has proposed a new method of call stack coverage that helps to overcome the traditional methods of reduction like static analysis based on line or statement coverage. It performed better in case of GUI testing where cross browser and event sequences needs to be handled. Call backs for event handler, multithreading, reflection etc. is well handled by call stack coverage. Another approach [28], uses program slicing technique for GUI Test Suite Reduction by identifying the redundant test cases. How can machine learning optimize the process of Test Suite Reduction in the context of GUI testing is a challenging question. Table 5 summarizes some of the latest work in this context.

Table 5. Latest Test Suite Reduction Approaches

Ref.	Latest Research Work
[29]	***Objective:*** Clustering to optimize test execution time ***Methodology:*** Cluster the similar test cases and select a single test case from the suite as a representative. ***ML Algorithm:*** k-means clustering ***Whether specific to GUI Testing:*** No ***Evaluation Criteria:*** Statement, Branch, Decision Coverage and Mutation Score. ***Results:*** Algorithm assures an average reduction of 82.2% having the same coverage and mutation score as the original test set.
[30]	***Objective:*** Identifying redundancy in test cases to optimize testing time. ***Methodology:*** Finds similarity within test cases based on coverage information. ***ML Algorithm:*** SVM, K-Nearest and Decision Tree. ***Whether specific to GUI Testing:*** No ***Evaluation Criteria:*** Performance is evaluated through error and accuracy metrics. ***Results:*** SVM outperforms with 71.43% accuracy.
[31]	***Objective:*** Identifying infeasible test cases to optimize testing resources. ***Methodology:*** Using SVM classifier to classify infeasible test cases. ***ML Algorithm:*** SVM and Induced Grammars. ***Whether specific to GUI Testing:*** Yes ***Evaluation Criteria:*** Percentage of test cases correctly classified w.r.t length of test cases. ***Results:*** It proved how induced grammar can show event sequences and constraint. Pairwise input extraction algorithm worked best in SVM.
[32]	***Objective:*** Reducing software regression testing cost. ***Methodology:*** Using Similarity based test cases clustering. ***ML Algorithm:*** K-means and K-means++ clustering. ***Whether specific to GUI Testing:*** Yes ***Evaluation Criteria:*** Size of Test Suite reduction and Fault Detection Loss. ***Results:*** Efficiently finds subsets of test cases with reduced test time in budget and adequate version.

Though, literature study retrieved more than 85 papers related to Test Suite Reduction, but it was found that very rare literature is present that matched proposed problem statement. It can thus be derived that, there is a vast area of scope for applying ML Techniques in this prospect.

4.3 Identification of Unreliable/Flaky test cases using ML

TCO based on reliability of test cases is the recent research dimension. Determining the quality of tests is in itself very crucial for testing accuracy. According to [33], approx. 73K of 1.6 M test failures per day is recorded at Google due to flaky tests. Authors in [34] has mentioned flakiness of test cases as unreliability in test case behavior i.e. sometimes it fails and sometimes it passes, without a change to the

underlying product or app. This in turn allows engineers to spend extra time identifying the issues. They proposed 'Probabilistic flakiness score (PFS)' to monitor the quality of test cases using Bayesian Inference Algorithm implemented in statistical modeling environment 'stan'. In [35] authors has provided a detailed analysis on issues, causes, costs and repairing strategies related to flaky tests. GUI test cases can be more flaky or brittle due to unpredictable changes over time and its dependency on application, environment and certain non-deterministic values [36]. Certain other techniques like Athena [37] and Odeneye [38] has been created to detect and handle test case flakiness. It's therefore important to monitor those tests that can reduce the quality of testing results. Table 6 illustrates some of the recent literature on Test Case Flakiness.

Table 6. Important research related to Test Case Flakiness

Category	***Contribution***	***Result***
Empirical Analysis on flakiness on 235 UI Test cases from 62 projects of both web and mobile projects [39]	Provided study on relationships between root causes and fixing strategies of flakiness	Root Cause identified: Async wait, Test Script Logic Issues, Test Runner API issue, Environment. Fixing Strategies: Disable Animation, Add Delay, Fix Await mechanism, Refactor Logic, Fix API Access, Change Library Version.
Identification and Fixing of the UI flaky tests in iTrust2 (Electronic Health Record System) [40]	Studied the impact of Web Driver, Wait Conditions, Hardware, Host Operating System and Effect of Restarting the web browser between tests	HtmlUnit yields fewer flaky tests than chrome. Thread waits provide lowest flakiness for both HtmlUnit and chrome, while explicit waits gives highest. Paper also contributes towards providing more stable and reliable teaching application.
Automatic fixing of flaky tests[41]	Targets problems of ordered dependent tests that are flaky. Recommends patches to fix flaky tests.	Evaluating iFixFlakies on 110 ordered dependent test from a public data set, showed that it can automatically recommend patches for 58 out of 110.
Identification and finding the root cause of flaky tests[42]	Log analysis tool 'RootFinder' that analyses log of runtime properties by finding differences in the logs of passing and failing runs.	Some of root causes analysed are Time, Randomness, Async Wait, Concurrency and Resource Leak. Paper also mentioned some of the research challenges in this area.
Identification of Flaky tests without rerunning the tests[43]	ML based approach that looks for test behaviour and predict flaky tests based on similarities in test behaviour.	FlakeFagger reported fewer false positives as compared to other classifiers.

5. RESEARCH GAPS

Studying and analyzing the available literature on GUI test cases in the context of ML, very less literature were retrieved. It has been at an attention of researchers since a decade, with CI/CD and Agile becoming more applicable. Papers specific in providing TCO w.r.t GUIs regression testing were around 20 only. Some of the key finding in terms of research gaps can be stated as follows.

- It has been observed that ML techniques has been generally applied in understanding the input sequences of GUI Android apps, Web Based Applications are not much explored.
- Optimizing the execution of GUI test cases for improving the test coverage seems to be tough and harder problem for researcher.
- Test Case Generation using event interaction graph or model based testing seems to be the best choice for researcher to analyze the structure and behavior of GUIs.
- Test Case Prioritization and Test Case Selection using ML has been done mostly for unit test cases.
- SVM, Clustering, Decision Trees, Gradient Boost and Reinforcement Learning algorithms were seen to dominate.
- Rare use of ensemble techniques has been seen for Test Case Optimization.
- Approach using white box testing data were used more in comparison to black box testing data.
- In the context of GUI testing, it was seen that very rare literature is present that contributes towards minimizing Test case flakiness through ML. Though, lots of literature are present in the context of test flakiness and ML, but for UI and specific to GUI test cases, there are very few.
- Generating and gathering data sets related to testing of GUIs are tough and tedious, which possess one of the greatest challenge for researchers.

- Systematic literature reviews and literature analysis on GUI regression testing is very rare and needs to be done more.
- Empirical papers are also very limited in this dimension.

6. CONCLUSION AND FUTURE SCOPE

This paper provides recent developments in the research of GUI functional regression testing in terms of Test Case Optimization mostly focusing in and around 5- 6 years. Reason behind selection of limited period was a steady progress of automation graph in these 5 years.Where,industries are struggling to keep up user's expectations in time-constrained software models. Research literature primarily providing an analysis of GUI Test cases were found very rare. This paper tried to structure the scenarios of Test Case Optimization putting some of the latest and important research papers in the context of GUI Test cases. In future, more detail systematic literature review can be carried out consisting of papers from past 20 years to see the growth. A mapping study can be also done to structure the scenarios of regression testing and GUI testing. Papers exploring DL, NLP and Computer Vision can also be considered in this context.

7. REFERENCES

[1] Ngah A., Munro M., AbdallahM,"An Overview of Regression Testing", Journal of Telecommunication, Electronic and Computer Engineering 9, pp. 45-49, 2017.

[2] Durelli V.H., Durelli R.S., Borges S.D., Endo A.T., Eler M.M., Dias D.R., Guimarães M.P, "Machine Learning Applied to Software Testing: A Systematic Mapping Study". IEEE Transactions on Reliability 68, pp 1189-1212, 2019.

[3] Nass M.D., Alégroth E., Feldt R, "Why many challenges with GUI test automation (will) remain", Inf. Softw. Technol. 138, 106625, 2021.

[4] "Infer", Available at: https://fbinfer.com/. Accessed on 20 February 2022.

[5] "Diffblue", Available at: https://www.diffblue.com/.Accessed on 21 March 2022.

[6] "SmartBear" Available at: https://smartbear.com/. Accessed on 21 March 2022.

[7] Chittimalli P.K., Harrold M.J,"Recomputing coverage information to assist regression testing". IEEE Transactions on Software Engineering 35(4): pp. 452–469, 2009.

[8] "softwaretestinghelp", Available at: https://www.softwaretestinghelp.com/. Accessed on 26 March 2022.

[9] Arora P.K. and Bhatia R. 2018. Agent-Based Regression Test Case Generation using Class Diagram, Use cases and Activity Diagram. In Procedia Computer Science 125: pp 747-753. ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.12.096 2018.

[10] Wetzlmaier T. and Ramler R. 2017. Hybrid monkey testing: enhancing automated GUI tests with random test generation. In Proceedings of the 8th ACM SIGSOFT International Workshop on Automated Software Testing.

[11] Kamal M.M., Darwish S.M. and ElfatatryA. 2019.Enhancing the Automation of GUI Testing. In Proceedings of the 2019 8th International Conference on Software and Information Engineering.

[12] Granda M.F., Gonzalez O.P., and Alba-Sarango B.2021.Towards a Model-Driven Testing Framework for GUI Test Cases Generation from User Stories. In Proceedings of16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021), pages 453-460.

[13] Memon A.M., Soffa M.L,"Regression testing of GUIs", SIGSOFT Softw. Eng. Notes 28, 5, pp 118–127, https://doi.org/10.1145/949952.940088, 2003.

[14] Pan M., Xu T., Pei Y., Li Z., Zhang T. and Li X," GUI-Guided Test Script Repair for Mobile Apps" IEEE Transactions on Software Engineering, 2020.

[15] Gao Z., Chen Z., Zou Y. andMemon A.M," SITAR: GUI Test Script Repair", IEEE Transactions on Software Engineering 42, pp 170-186, 2016.

[16] Engström E.,Runeson P. and Skoglund M," A systematic review on regression test selection techniques," Inf. Softw. Technol 52, pp 14-30, 2010.

[17] Kazmi R., Jawawi D.N., Mohamad R. andGhani I," Effective Regression Test Case Selection", ACM Computing Surveys (CSUR) 50, pp- 1 – 32, 2017.

[18] Magalhães C., Mota A. andMomente L," UI Test case prioritization on an industrial setting: A search for the best criteria." Software Quality Journal 29, pp 381–403, https://doi.org/10.1007/s11219-021-09549-y 2021.

[19] Machalica M., Samylkin A., Porth M. and Chandra S. 2019.Predictive Test Selection. In proceedings of the IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSESEIP), pp 91-100.

[20] "launchableinc". Available at: https://www.launchableinc.com/what-is-test-impact-analysis. Accessed on 10 March 2022.

[21] Maitrikul C. and Limpiyakorn Y., "GUI Test Case Prioritization using Social Network Analysis", Journal of Physics. IOP Publishing. DOI: 10.1088/1742-6596/1619/1/012020,2020.

[22] He Z.W. and Bai C.G. 2015. GUI Test Case Prioritization by State-Coverage Criterion. In 2015 IEEE/ACM 10th International Workshop on Automation of Software Test, pp 18-22, DOI: 10.1109/AST.2015.11.

[23] Sun W., Gao Z., Yang W., Fang C. and Chen Z. 2013 .Multi-objective test case prioritization for GUI applications In proceedings of the 28th Annual ACM Symposium on Applied Computing.

[24] Yu Z., Fahid F.M., Menzies T., Rothermel G., Patrick K. and Cherian S.2019.TERMINATOR: better automated UI test case prioritization.In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.

[25] Coppola R., Morisio M. and Torchiano M. 2017. Scripted GUI Testing of Android Apps: A Study on Diffusion, Evolution and Fragility. In Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering.

[26] Nguyen V. and Le B," RLTCP: A reinforcement learning approach to prioritizing automated user interface tests", Information and Software Technology, Volume 136, https://doi.org/10.1016/j.infsof.2021.106574, 2021.

[27] McMaster S. and Memon A.M. 2005. Call stack coverage for test suite reduction. In proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), pp. 539-548,doi: 10.1109/ICSM.2005.29.

[28] Arlt S., Podelski A. andWehrle M. 2014. Reducing GUI test suites via program slicing. In proceedings of ISSTA.

[29] Chetouane N., Wotawa F., Felbinger H. and Nica M. 2020. On Using k-means Clustering for Test Suite Reduction. In proceedings of 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp 380-385.

[30] Saputra M.C. and Katayama T. 2020. Code Coverage Similarity Measurement Using Machine Learning for Test Cases Minimization. In proceedings of IEEE 9th Global Conference on Consumer Electronics (GCCE), pp 287-291, doi: 10.1109/GCCE50665.2020.9291990.

[31] Gove R.J. and Faytong J. 2011. Identifying Infeasible GUI Test Cases Using Support Vector Machines and Induced Grammars. In IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, pp 202-211.

[32] Cruciani E., Miranda B., Verdecchia R. and Bertolino A. 2019. Scalable Approaches for Test Suite Reduction. In proceedings of 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp: 419-429.

[33] Memon A.M., Gao Z., Nguyen B., Dhanda S., Nickell E., Siemborski R. and Micco J. 2017. Taming Google-Scale Continuous Testing. In IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp 233-242.

[34] "Engineering at Meta". Available at: https://engineering.fb.com/2020/12/10/developertools/probabilistic-flakiness/. Accessed on 20 February 2022.

[35] Parry O., Kapfhammer G.M., Hilton M.C. and McMinn P., "A Survey of Flaky Tests", ACM Trans. Softw. Eng. Methodol. 31, 17:1-17:74, 2021.

[36] Dix A, 1990. Non-determinism as a paradigm for understanding the user interface. In E. H. Harrison, editors. Formal Methods in Human Computer Interaction. Cambridge University Press, pp 97-127.

[37] "Dropbox.Tech". Available at: https://dropbox.tech/infrastructure/athena-our-automatedbuild-health-management-system. Accessed on 26 March 2022.

[38] Spotify "R&D Engineering". Available at: https://engineering.atspotify.com/2019/11/testflakiness-methods-for-identifying-and-dealingwith-flaky-tests/. Accessed on 28 March 2022.

[39] Romano A., Song Z., Grandhi S., Yang W. and Wang W.2021. An Empirical Analysis of UI-Based Flaky Tests. In proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp 1585-1597.

[40] Marshall K.P., Horton E., Heckman S. andStolee K. 2019. Wait, Wait. No, Tell Me. Analyzing Selenium Configuration Effects on Test Flakiness. In proceedings of the IEEE/ACM 14th International Workshop on Automation of Software Test (AST), pp 7-13, doi: 10.1109/AST.2019.000.

[41] Shi A., Lam W., Oei R., Xei T. and Marinov D. 2019.iFixFlakies: A framework for automatically fixing order dependent flaky tests. In Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19) New York, doi:https://doi.org/10.1145/3338906.3338925.

[42] Lam W., Godefroid P., Nath S., Santhiar A. and Thummalapenta S. 2019. Root Causing Flaky Tests in a Large-Scale Industrial Setting. In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis.

[43] Alshammari A., Morris C., Hilton M. and Bell J. 2021. FlakeFlagger: Predicting Flakiness without Rerunning Tests.In Proceedings of the 43rdInternational Conference on Software Engineering (ICSE'21).